European Journal of Control 78 (2024) 101019

Contents lists available at ScienceDirect

European Journal of Control

journal homepage: www.sciencedirect.com/journal/european-journal-of-control

Revisiting reachability-driven explicit MPC for embedded control

Juraj Holaza, Peter Bakaráč, Juraj Oravec *

Institute of Information Engineering, Automation, and Mathematics, Faculty of Chemical and Food Technology, Slovak University of Technology in Bratislava, Radlinskeho 9, Bratislava, SK812-37, Slovak Republic

ARTICLE INFO

Recommended by T. Parisini

Keywords:
Model predictive control
Complexity reduction
Reachability sets
Parametric optimization
Binary encoding

ABSTRACT

The real-time implementation of the explicit MPC suffers from the evaluation of the, potentially large, lookup table. The paper revisits the original approach and presents an efficient reachability-sets-driven-based explicit MPC method addressing this issue by splitting the look-up table into the set of the "relevant" subsets. Simultaneously, effective binary encoding is introduced to minimize the runtimes and the memory footprint. Further acceleration is achieved by introducing the "smart" order of the considered critical regions. Then, the significant real-time complexity reduction is ensured by online pruning and traversing the sorted lookup table associated with the optimal control law evaluation. Technically, the number of critical regions to be explored is reduced and the order is redefined to accelerate the point location problem and minimize the computational effort. While the optimality of the control law is still preserved, the cost that we need to pay for the accelerated point location problem lies in an additional offline computation effort and a minor increase in memory requirements of the underlying controller. The benefits of the proposed method are demonstrated using an extensive case study. The complexity reduction strategy was investigated on two fast-dynamic benchmark systems and the computational burden was analyzed by implementing the designed controllers on an embedded hardware.

1. Introduction

Model predictive control (MPC) represents an advanced control strategy that enjoys worldwide success in both academia and industry (Mayne, 2014; Morato, Normey-Rico, & Sename, 2020; Qin & Badgwell, 2003). This popularity stems from its robustness and versatility in embedding all essential physical, economical, and environmental restrictions of the system directly into the optimization problem while still maximizing profit via performing predictions of the controlled system evolution within a finite time horizon (Borrelli, 2017). One of the main drawbacks of the MPC policy is the computational complexity of the underlying optimization problem. Specifically, in order to maintain stability, optimality, and constraint satisfaction, the MPC optimization problem has to be solved at each sample instant. This criterion can be quickly jeopardized, especially for systems with rapid sampling periods or when aiming to implement MPC policy on low-level hardware with strictly limited computation resources (McInerney, Constantinides, & Kerrigan, 2018).

Explicit Model Predictive Control (EMPC) methodology introduces parametric programming to offline pre-calculate the entire MPC optimization problem, i.e., a mapping between all feasible initial conditions and associated optimal control actions is being constructed (Bemporad, Morari, Dua, & Pistikopoulos, 2002). It was shown that for a wide range of MPC formulations, the explicit solution takes the form of a piece-wise affine (PWA) control law defined over a polytopic support (Borrelli, 2017). Subsequently, online computation of optimal control actions is then restricted only to a mere function evaluation that can be carried out on an arbitrary control hardware without the necessity of employing any optimization solver, i.e., leading to a library-free code. This makes EMPC a suitable candidate for a straightforward, fast, and easily certifiable real-time optimal control strategy (Oberdieck, Diangelakis, Nascu, Papathanasiou, Sun, Avraamidou, & Pistikopoulos, 2016).

It is known that EMPC has its shortcomings. Firstly, the offline construction of the PWA control law is quite computationally exhausting, and even though effective approaches were developed (Borrelli, Baotić, Pekar, & Stewart, 2010; Gupta, Bhartiya, & Nataraj, 2011; Herceg, Jones, Kvasnica, & Morari, 2015; Mitze, Kvasnica, & Mönnigmann, 2023; Oberdieck, Diangelakis, & Pistikopoulos, 2017), the applicability of EMPC methodology is still restricted only to small or moderate-size problems. Secondly, the complexity of the resulting explicit PWA control law, usually expressed in the number of critical regions defining the polytopic support of the PWA function, grows exponentially with the problem size. To keep the memory footprint of an explicit solution still trackable, various memory reduction techniques were proposed. Generally, they can be split into methods that maintain

* Corresponding author.
E-mail addresses: juraj.holaza@stuba.sk (J. Holaza), peter.bakarac@stuba.sk (P. Bakaráč), juraj.oravec@stuba.sk (J. Oravec).

https://doi.org/10.1016/j.ejcon.2024.101019
Received 15 December 2023; Received in revised form 19 March 2024; Accepted 11 May 2024
Available online 17 May 2024

optimality (Geyer, Torrisi, & Morari, 2008; Holaza, 2012; Kvasnica & Fikar, 2010, 2012; Mitze et al., 2023) or those that impose a sub-optimality (Bakaráč, Holaza, Klaučo, Kalúz, Löfberg, & Kvasnica, 2018; Holaza, Takács, Kvasnica, Di Cairano, 2015a; Jones & Morari, 2010; Kvasnica, Löfberg, & Fikar, 2011), which is usually being minimized. The third drawback of EMPC policies is the online (real-time) evaluation that is usually restricted to identifying the optimal affine piece, of the PWA function, defined over a critical region containing the current state measurement. This set-membership task is also referred to as the *point location problem* and it is of paramount importance to mitigate its runtime to meet the hardware computation limitations and secondly to minimize the energy consumption of the control operation in case of the battery supplied control units, such as unmanned aerial vehicles (UAVs)/drones, etc. Yan, Zhang, Chen, and Shi (2023). The goal of this paper is to address this third issue.

The point location problem is an important task that attracted a lot of attention from numerous researchers and one can find various advanced acceleration techniques in the literature. The most common approach is the *sequential search* algorithm that traverses through critical regions of the polytopic support until a critical region containing the current state is found. Then, the associated affine piece of the control law is evaluated to obtain the optimal control action that is applied to the controlled system. This approach was improved by, e.g., building binary trees (Bayat, Johansen, & Jalali, 2012; Fuchs, Jones, & Morari, 2010; Johansen & Grancharova, 2003; Tøndel, Johansen, & Bemporad, 2003; Zhang & Xiu, 2018), graph traversal methods (Herceg, Mariéthoz, & Morari, 2013; Jafargholi, Peyrl, Zanarini, Herceg, & Mariethoz, 2014; Wang, Jones, & Maciejowski, 2007), evaluating a value function (Baotić, Borrelli, Bemporad, & Morari, 2008; Borrelli, Baotić, Bemporad, & Morari, 2001; Nguyen, 2015), creating a bounded box for each critical region (Bemporad, Filippi, & Torrisi, 2004; Christophersen, Kvasnica, Jones, & Morari, 2007), parallel computation (Oravec, Jiang, Houska, & Kvasnica, 2017), lattice piecewise affine representation (Wen, Ma, & Ydstie, 2009; Xu, 2021), hash tables (Bayat, Johansen, & Jalali, 2011; Changizi, Salahshoor, & Siahi, 2023; Zhang, Xiu, Xie, & Hu, 2016), utilizing a low precision arithmetic (Suardi, Longo, Kerrigan, & Constantinides, 2016), and, last but not least, via sorting indices of critical regions (Holaza, Oravec, Kvasnica, Dyrska, Mönnigmann, & Fikar, 2020). Needless to say, the point location problem is generally proportional to the number of regions of the PWA control law. Hence, all aforementioned memory reduction techniques can be also used to significantly improve the online evaluation time of explicit MPC policies (even on top of online acceleration techniques).

From the applicability point of view, all point location problem approaches have their advantages and limitations. Generally, the sequential search algorithm is the slowest method in the worst-case scenario, but this method can be efficiently applied to any explicit controller. On the other hand, binary search tree schemes (Bayat et al., 2012; Fuchs et al., 2010; Johansen & Grancharova, 2003; Tøndel et al., 2003; Zhang & Xiu, 2018) represent one of the fastest and most memory-efficient approaches. However, deriving well-balanced trees for higher-dimensional controllers is usually a hardly tractable task. Next, the value function evaluation methods (Baotić et al., 2008; Borrelli et al., 2001; Nguyen, 2015) exhibit a memory-efficient structure with a fixed online evaluation time, as the same number of piecewise affine (PWA) functions need to be considered at each sample instant. On the contrary, online computation of the sorted sequential search algorithm in Holaza et al. (2020) converges with time to exploration only a single region, hence the computational effort is minimal around steady operation. It should be emphasized that some methods can be used in conjunction with other approaches to unlock even further efficiency. For example, the sequential search can be enhanced with the region sorting (Holaza et al., 2020), bounded box evaluation (Christophersen et al., 2007), and two-level structure (Zhang et al., 2016) techniques simultaneously. Likewise, the hash table (Bayat et al., 2011) and the binary search tree (Tøndel et al., 2003) were merged in Zhang and Xiu (2018) to ease the offline computation burden, and still one can employ (Changizi et al., 2023), for example, for further complexity reduction. Needless to say, all methods differ in how they trade the online evaluation effort of the explicit controller with offline computation, memory requirements, and, finally, optimality of the control actions. These methods can be efficient for certain problems but prohibitive for others, i.e., these methods can be compatible with other approaches, or they can have a specific structure that does not allow the application of other techniques. therefore, it is based on the specific expert knowledge, or the advanced engineer heuristics, to properly choose the appropriate point location approaches that suit the specific hardware/plant needs. The main goal of this paper is to show that the widely-used sequential search algorithm can also be accelerated by exploiting the reachability analyses and that the resulting method can be built on top of other previously mentioned techniques.

Reachability analysis (Borrelli, 2017; Mayne, Seron, & Raković, 2005) represents a powerful tool that is utilized in various techniques, e.g., to verify properties of an MPC policies (Bemporad, Torrisi, & Morari, 2000; Holaza, Takács, Kvasnica, & Di Cairano, 2015b). In Kvasnica, Bakaráč, and Klaučo (2019), less complex EMPC feedback laws are designed using the reachability analysis by initial reduction of the admissible set of the initial conditions of the system states. Even though the construction of exact reachable sets is generally computationally intensive for higher dimensions (Bird, Jain, Pangborn, & Koeln, 2022; Stursberg & Krogh, 2003) a workaround can be found by employing their approximate counterparts (Althoff, Frehse, & Girard, 2021; Bemporad et al., 2004), which are easier to construct, but lead to more conservative results. Reachability, in the context of the point location problem, was introduced in Spjøtvold, Raković, Tøndel, and Johansen (2006) where a list of indices of all reachable regions was computed for each region within the polyhedral domain. It was pointed out that this method can dramatically reduce the online computation of optimal control actions as the point location problem needs to explore only a small subset of possible candidates. The increasing computation burden of reachable sets was addressed in Sui, Feng, and Hovd (2008). Specifically, it was shown here that the construction of an exact reachable set can be reduced to perform only the reachability of a point and determine its application boundary. By employing this method one can easily reduce the offline computation burden, hence increasing the applicability of the reachability technique for a wider range of systems. The problem, however, is that due to the conservative selection of both the reachable point and its application domain, the final list of possible candidates tends to be larger compared to the list given by the exact approach. Motivated by these results, we aim to propose a novel technique that can be used on top of any of these two mentioned methods and can even further accelerate the point location problem and decrease the memory footprint of the constructed reachable list.

In this paper, we revisit the original work (Spjøtvold et al., 2006) that exploits information from the reachability analysis to reduce the complexity of the point location problem. While the proposed results are arguably not unexpected, to the best authors' knowledge, the presented results, formulated remarks, and detailed analysis are lacking in the existing literature. Specifically, for each critical region of the polytopic support, one can offline determine indices of all one-step forward reachable regions and store them in a *list*. First, compared to the original work (Spjøtvold et al., 2006), this paper formulates several rigorous theoretical contributions and provides much deeper insight into the proposed acceleration method. Next, we directly extend the results of Spjøtvold et al. (2006) by pushing the original idea towards its memory-efficient implementation and evaluation on the embedded hardware using the binary representation of the list. Finally, we introduce significant acceleration by the novel *sorted* list, while preserving the optimality of the closed-loop control. We point out that as the sorted list is constructed offline, this additional acceleration layer does not negatively affect either the average online runtimes or the

worst-case evaluation. Using the experimentally collected data based on the laboratory implementation on the embedded hardware, we show that such a sorted list accelerates the online evaluation by interesting factors.

The rest of the paper has the following structure. Section 2 formulates the problem of real-time evaluation of optimal control action. Section 3 introduces the reachability sets-driven method speeding up the real-time evaluation of optimal control action (Section 3.1) and its robustification subject to the impact of the disturbances (Section 3.2). Implementation details are discussed in Section 4. Section 5 presents the further novel acceleration method based on the reachability analysis. The benefits of the proposed methods are analyzed in Section 6 using an illustrative benchmark (Section 6.1) and two benchmark systems with fast-dynamics (Sections 6.2 and 6.3), followed by the main conclusions summarized in Section 7.

Notation

Throughout the paper, we use a conventional notation. Denote $\mathbb{R}^n$ and $\mathbb{R}^{n\times m}$ the set of real-valued n-dimensional vectors and $n\times m$ matrices, respectively.
Denote $\mathbb{N}^n$, $\mathbb{N}^n_+$, and $\mathbb{N}^n_{++}$ the sets of arbitrary, non-negative, and positive integer-valued n-dimensional vectors, respectively.
For vector $x \in \mathbb{R}^{n_x}$, square matrix $Q \in \mathbb{R}^{n_x\times n_x}$, and $p \in \{1, 2, \infty\}$, the considered p-norms have, respectively, the form: $\|x_k\|_Q^1 \triangleq \sum |Qx|$, $\|x_k\|_Q^2 \triangleq x^\top Qx$, $\|x\|_Q^\infty \triangleq \max |Qx|$, where $|Qx| \in \mathbb{R}^{n_x}$ is a vector.
Denote $|\cdot|$ to be a cardinality of vector $w \in \mathbb{R}^n$, i.e., the number of its elements n, diag$(\cdot)$ to be a diagonal matrix, and $\mathbf{1}_n$ to represent a vector of ones of size n.
A polytope is a closed and bounded convex set defined by a finite number of half-spaces.

2. Problem statement

Assume a state-space representation of a discretized linear time-invariant (LTI) system in the form

$$x(t + T_s) = Ax(t) + Bu(t), \tag{1}$$

where $t \in \mathbb{R}$ is the time variable, T_s is the sampling time, $x(t) \in \mathbb{R}^{n_x}$ is the state vector, $u(t) \in \mathbb{R}^{n_u}$ is the input vector, $A \in \mathbb{R}^{n_x\times n_x}$ is system state matrix, $B \in \mathbb{R}^{n_x\times n_u}$ is input matrix. Assume that the system (1) is subjected to

$$x(t) \in \mathcal{X},\ u(t) \in \mathcal{U}, \tag{2}$$

where $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ and $\mathcal{U} \subseteq \mathbb{R}^{n_u}$.

Assumption 2.1. Assume, the matrix pair (A, B) is stabilizable and $\mathcal{X}$, $\mathcal{U}$ are non-empty compact full-dimensional polyhedral sets containing the origin in their strict interiors.

To control the system in (1) asymptotically to the origin, while providing recursive feasibility of the constraints in (2), one constructs a constrained finite-time optimal control (CFTOC) problem as

$$\min_{\substack{u(k),\, x(k+1),\\ k=0,\dots,N-1}} \|x_N\|^p_{Q_N} + \sum_{k=0}^{N-1} \left(\|x_k\|^p_{Q_x} + \|u_k\|^p_{Q_u} \right) \tag{3a}$$

$$\text{s.t. } x(k+1) = Ax(k) + Bu(k), \tag{3b}$$

$$x(k) \in \mathcal{X}, \quad u(k) \in \mathcal{U}, \tag{3c}$$

$$x(N) \in \mathcal{X}_N, \tag{3d}$$

$$x(0) = x(t), \tag{3e}$$

where $p \in \{1, 2, \infty\}$ in (3a) denotes 1-, 2-, or ∞-norm of the minimized cost function. The CFTOC problem with model, state, and input constraints in (3b)–(3c) are enforced for all prediction steps $k = 0, \dots, N-1$. Denote $N \in \mathbb{N}_{++}$ as the prediction horizon, $Q_N \succ 0$, $Q_u \succ 0$, and $Q_x \succeq 0$ to be weighting matrices of appropriate dimensions. Moreover, for the set of feasible initial conditions $\mathcal{F}$, define the terminal set $\mathcal{X}_N \subseteq \mathcal{F} \subseteq \mathcal{X} \subset \mathbb{R}^{n_x}$ as a polytopic positive invariant set containing the origin in its strict interior. If $p = 2$ in (3a), then the CFTOC problem (3) is formulated in the form of quadratic programming (QP), otherwise, for $p \in \{1, \infty\}$ in (3a), then CFTOC problem (3) is formulated in the form of linear programming (LP). The resulting CFTOC problem is solved in each control step for a particular feasible state measurement (estimate) $x(t) \in \mathcal{F}$, via various state-of-the-art solvers, yielding a sequence of optimal control inputs $U^\star = [u^\star(0)^\top, \dots, u^\star(N-1)^\top]^\top$. By applying the receding horizon policy, only the first input of this sequence $u(t) = u^\star(0)$ is applied to the system at the given time step t.

Algorithm 1 Sequential search point location problem.

```
Inputs: x(t), list of indices I, feedback law κ(x(t))
Output: the optimal control action u⋆(t)
for i ∈ I do
    if x(t) ∈ R_i then
        i⋆ ← i
        break
    end if
end for
Execute u⋆(t) ← κ(x(t)) for i⋆ per (4)
```

Lemma 2.1 (*Borrelli, Bemporad, & Morari, 2017*). *Parametric solution to (3) is a continuous piecewise affine (PWA) function $u^\star(0) = \kappa(x(t))$ mapping vector of initial states $x(t)$ onto the vector of optimal control action $\kappa : \mathcal{F} \mapsto \mathbb{R}^{n_u}$. This function is given as*

$$\kappa(x(t)) \triangleq F_i x(t) + g_i \quad \text{if} \quad x(t) \in \mathcal{R}_i, \tag{4}$$

with $F_i \in \mathbb{R}^{n_u\times n_x}$, $g_i \in \mathbb{R}^{n_u}$, $\forall i \in \{1, 2, \dots, M\}$.

The polytopic partition $\mathcal{F}$ satisfies following properties:

(a) $\cup_{i=1}^M \mathcal{R}_i = \mathcal{F}$,
(b) $\mathcal{F} = \{x(t) \mid \exists u : \text{(3c) holds}\}$,
(c) $\text{int}(\mathcal{R}_i) \cap \text{int}(\mathcal{R}_j) = \emptyset$, $\forall i \neq j$,

where $\text{int}(\mathcal{R}_i)$ *denotes interior of the ith critical region.*

The real-time evaluation of $\kappa(x(t))$ in (4) is done in two steps. Firstly, we need to determine an index $i^\star \in \mathcal{I} = \{1, \dots, M\}$ for which $x(t) \in \mathcal{R}_{i^\star}$ holds. Without loss of generality, we determine $i^\star$ considering the commonly used *sequential search* algorithm to demonstrate the main idea of the paper, see Algorithm 1. Note, the algorithms for *binary search trees* can be adopted in an analogous way.

In the second step, we compute the affine expression $\kappa(x(t)) = F_{i^\star} x(t) + g_{i^\star}$ as in (4). Since the second step involves only the evaluation of a simple affine expression, the main time-demanding effort is required in the first step, especially for large $M \gg 10^3$ and/or high dimension of the parametric space n_x.

Therefore, this paper investigates the possibilities to accelerate the point location problem in the Algorithm 1, i.e., to decrease the number of explored critical regions, by providing a reduced, possibly sorted, and efficiently stored list of indices.

3. Reachability sets driven explicit MPC

Even with an explicit formulation (4) in hand, i.e., a set of affine expressions defined over a feasible polytopic partition $\mathcal{F}$ containing M critical regions $\mathcal{R}_i$, the online evaluation can be still troublesome. Specifically, in the worst-case, the sequential search Algorithm 1 traverses through the list of *all* possible indices $\mathcal{I} = \{1, \dots, M\}$ until the active index $i^\star \leq M$ is found for which $x(t) \in \mathcal{R}_{i^\star}$. Note, the point location problem is significantly accelerated if the total number of considered critical regions M is reduced or if the index $\mathcal{I}$ is rearranged.

In this paper, we aim to target the second option, i.e., the basic idea is to replace the $\mathcal{I}$ with a new list of indices $\widetilde{\mathcal{I}}$ that exploits information of a reachability analysis. This section revisits the original work (Spjøtvold et al., 2006) and formulates several rigorous theoretical contributions and provides deeper insight into the acceleration method based on the reachability analysis. It will be shown that $\widetilde{\mathcal{I}}$ accelerates the search for $i^\star$ by an interesting factor,[1] however at the cost of an increased memory footprint. This drawback is further addressed in Section 4, where an approach on how to efficiently store the structure of $\widetilde{\mathcal{I}}$ is presented.

Remark 3.1 (*Application Range*)**.** The proposed method is not limited to the MPC formulation in (3) as the control law in (4) can be formulated by using a wide variety of MPC configurations including robust/tube MPCs; a reference tracking problem, trajectory preview in (3a); hybrid/PWA system model in (3b); slew-rate constraints in (3c), etc. See Borrelli et al. (2017), and references therein, for the particular CFTOC problem formulations.

3.1. Reachability analysis of MPC control laws

In this section, we show how the reachability analysis enables a significant reduction of the point location problem associated with the execution of (4). For a given LTI system in (1), the one-step forward reachable set of $\mathcal{X}$ is defined as in Borrelli et al. (2017, Def. 10.4.):

$$\text{Reach}(\mathcal{X}) = \{Ax(t) + Bu(t) \,|\, \exists x(t) \in \mathcal{X}, \exists u(t) \in \mathcal{U}\}. \tag{5}$$

Technically, the reachability set represents all states from $\mathcal{X}$ that are mapped into the set Reach($\mathcal{X}$) under the map of system in (1) by applying any feasible control input from $\mathcal{U}$. If Assumption 2.1 holds, i.e., if $\mathcal{X}$ and $\mathcal{U}$ are polytopes, then also the Reach($\mathcal{X}$) is a polytope.

Consider that at the time step $(t+T_s)$ we need to evaluate $\kappa(x(t+T_s))$ in (4), hence to search for the active index $i^\star(t+T_s)$ per Algorithm 1 and to calculate the associated affine expression $F_{i^\star(t+T_s)}x(t+T_s) + g_{i^\star(t+T_s)}$. Moreover, assume that we have stored the active index at the previous time frame, i.e., $i^\star(t)$, for which $x(t) \in \mathcal{R}_{i^\star(t)}$ was satisfied, and the feedback law $u^\star(t) = F_{i^\star(t)}x(t) + g_{i^\star(t)}$ was applied to the system.

By plugging the optimal control law $u^\star(t)$ into the nominal model dynamics $Ax(t)+Bu^\star(t)$ we obtain an autonomous representation of the system into the form:

$$x(t+T_s) = (A + BF_{i^\star(t)})x(t) + Bg_{i^\star(t)}. \tag{6}$$

Next, we know that our initial state conditions are restricted only to the region $\mathcal{R}_{i^\star(t)}$, we have that $x(t) \in \mathcal{X}$ can be replaced by $x(t) \in \mathcal{R}_{i^\star(t)}$. Finally, as the control law $u^\star(t)$ was computed such that $u^\star(t) \in \mathcal{U}$ if $x(t) \in \mathcal{R}_{i^\star(t)}$, then the input constraint $u(t) \in \mathcal{U}$ can be omitted. With this information in hand, we can restate the reachability set in (5) as

$$\text{Reach}(\mathcal{X}) = \{(A + BF_{i^\star(t)})x(t) + Bg_{i^\star(t)} \,|\, \exists x(t) \in \mathcal{R}_{i^\star(t)}\}. \tag{7}$$

As a consequence, the construction of reachable sets is formulated as a simple affine mapping of a polytope $\mathcal{R}_{i^\star(t)}$.

Definition 3.1 (*List of Reachable Regions*)**.** Let $\widetilde{\mathcal{I}}_i \subseteq \mathcal{I} \subset \mathbb{N}^M_{++}$, $\forall i \in \{1, 2, \dots, M\}$ be a list of positive integers corresponding to the indices of the critical regions $\mathcal{R}_j$ reachable from the critical region $\mathcal{R}_i$ as in (5), such that Reach($\mathcal{R}_i$) $\subseteq \cup_{j\in\widetilde{\mathcal{I}}_i}\mathcal{R}_j$ holds.

The set of reachable critical regions $\widetilde{\mathcal{I}}_i$, $\forall i = \{1, 2, \dots, M\}$ are constructed by the reachability analysis of an explicit MPC policy according to Algorithm 2. Specifically, consider an explicit feedback policy $\kappa(\cdot)$ as in (4) that is defined over a polytopic partition $\mathcal{P} = \cup_{i=1}^M \mathcal{R}_i$. Let us now assume a fixed critical region $\mathcal{R}_i$ for which we compute a reachable set Reach($\mathcal{R}_i$) per (5). This way we have found a set of all states where the $\kappa(\cdot)$ will drive the system (1) within one sample period T_s. Now we aim to determine which critical regions of $\kappa(\cdot)$ have a non-empty intersection with the set Reach($\mathcal{R}_i$). To do this, we need to traverse through the entire partition of $\kappa(\cdot)$ and find non-empty intersections of the reachable set and all critical regions $\mathcal{R}_j$ with $j \in \{1, \dots, M\}$.

Information if the intersection Reach($\mathcal{R}_i$) $\cap\, \mathcal{R}_j$ is (not) an empty set is stored in the list $\widetilde{\mathcal{I}}_i$, at its kth position, i.e., assigned into $\widetilde{\mathcal{I}}_{i,k}$, see Step 9 in Algorithm 2. As the consequence, the presence of some index $j \in \{1, 2, \dots, M\}$ in the list $\widetilde{\mathcal{I}}_i$ confirms that the critical region $\mathcal{R}_j$ is reachable from $\mathcal{R}_i$ within one sample step T_s, under the map (1), and by using a control action from $\mathcal{U}$. On the other hand, if some index $l \in \{1, 2, \dots, M\}$ is not included in the list $\widetilde{\mathcal{I}}_i$, then $\mathcal{R}_l$ is not reachable from $\mathcal{R}_i$ per (5). Finally, by iterating through all critical regions $\mathcal{R}_i$ with $i \in \{1, \dots, M\}$ we construct the set of M lists $\widetilde{\mathcal{I}}_i$ that concludes our reachability analysis.

Algorithm 2 Reachability analysis of $\kappa(\cdot)$ (Spjøtvold et al., 2006).

1: **Inputs:** polyhedral partition $\mathcal{P} = \cup_{i=1}^M \mathcal{R}_i$ of $\kappa(\cdot)$, list of indices $\mathcal{I}$
2: **Output:** M lists of reachable critical regions $\widetilde{\mathcal{I}}_i$, $\forall i = \{1, 2, \dots, M\}$
3: **for** $i \in \mathcal{I}$ **do**
4: define $k \leftarrow 1$
5: define $\widetilde{\mathcal{I}}_i \leftarrow \varnothing$
6: compute Reach($\mathcal{R}_i$) per (7)
7: **for** $j \in \{1, \dots, M\}$ **do**
8: **if** Reach($\mathcal{R}_i$) $\cap\, \mathcal{R}_j \neq \varnothing$ **then**
9: update $\widetilde{\mathcal{I}}_{i,k} \leftarrow j$
10: update $k \leftarrow k + 1$
11: **end if**
12: **end for**
13: **end for**

Remark 3.2 (*Lower Dimensional Intersections*)**.** From the implementation point of view of the Algorithm 2, the degeneracy of a polytope has to be addressed. Generally, Reach($\mathcal{R}_i$) $\cap\, \mathcal{R}_j$ can return a lower dimensional polytope that represents e.g. a facet or a vertex of $\mathcal{R}_j$. To stress this issue, let us consider Reach($\mathcal{R}_i$) = $\mathcal{R}_k$. The Algorithm 2 would include to the respective set $\widetilde{\mathcal{I}}_i$ not only the index k, but also all indices of its neighboring regions. This is caused by the definition of explicit MPC policy in (4) as each region of $\mathcal{P}$ is a polytope, i.e., a closed set. Needless to say, these lower-dimensional intersected polytopes can be omitted without loss of generality.

Remark 3.3 (*Lower Dimensional Reachable Sets*)**.** If the reachable set Reach($\mathcal{R}_i$) is a degenerate lower dimensional polytope, then we have to consider all intersected polytopes, i.e. even the degenerate ones. The argument behind this is straightforward. If Reach($\mathcal{R}_i$) is a lower dimensional polytope then also the intersection with the polytopic partition $\mathcal{P}$ returns only a set of lower dimensional polytopes. Since these intersected polytopes are not included in other full-dimensional polytopes we need to store them.

Lemma 3.4 (*Stability and Recursive Feasibility*)**.** *Given system in (1), CFTOC problem in (3) leading to the optimal control law in (4), $x(t-T_s) \in \mathcal{R}_i$, and $\widetilde{\mathcal{I}}_i$ according Definition 3.1. Replacing the full set of indices $\mathcal{I}$ by the rearranged set of indices $\widetilde{\mathcal{I}}_i$ into evaluating optimal control law according to Algorithm 1 preserves the recursive feasibility w.r.t. constraints in (2). Moreover, if the solution of the CFTOC problem in (3) ensures the asymptotic stability of the system in (1) for the set of indices $\mathcal{I}$, then the asymptotic stability is preserved for the rearranged set of indices $\widetilde{\mathcal{I}}$ as well.*

Proof. First, we prove the recursive feasibility. It follows from (5) that for any $\mathcal{R}_i \subset \mathcal{P}$ and $x(t-T_s) \in \mathcal{R}_i$ leads to $(Ax(t) + Bu(t)) \in \text{Reach}(\mathcal{R}_i) \subseteq \mathcal{P}$. Then, by Definition 3.1, $\widetilde{\mathcal{I}}_i \subseteq \mathcal{I}$ is such that $\cup_{j\in\widetilde{\mathcal{I}}_i}\mathcal{R}_j \supseteq \text{Reach}(\mathcal{R}_i)$. As the consequence, $(Ax(t) + Bu(t)) \in \text{Reach}(\mathcal{R}_i) \Rightarrow (Ax(t) + Bu(t)) \in$

[1] The acceleration rate is problem-dependent and its estimation is under further research.

$\cup_{j\in\widetilde{\mathcal{I}}_i}\mathcal{R}_j$ for any $x(t)\in\mathcal{F}$, and for $\forall u(t)\in\mathcal{U}$. Next, the proof of the asymptotic stability straightforwardly follows from the formulation of CFTOC in (3). If the feasible stabilizable solution of (3) exists for any $x(t)\in\mathcal{F}$ then the asymptotic stability is preserved for all upcoming states $(Ax(t)+Bu(t))$ as the consequence of the recursive feasibility. □

Under a mild assumption that the PWA control law is continuous (see Lemma 2.1), we would like to emphasize that the proposed acceleration method does not lead to any performance loss.

Corollary 3.4.1 (*Preserved Optimality of the Control Law*)**.** *As the consequence of Lemma 3.4, the method proposed in Algorithm 3 returns the same index i of the PWA function in (4) as Algorithm 1. Hence, the evaluation of this specific affine control law returns the same optimal control action $\kappa(x(t))$ in (4) of CFTOC problem in (3).*

Proof. The proof of Corollary 3.4.1 straightforward follows from the equivalence of the solutions of the point location problem evaluated by Algorithms 1, 3, see Spjøtvold et al. (2006). □

Moreover, as the same optimal control action is found, the closed-loop profiles are also the same by design. The only difference between the trajectories can occur due to the presence of the random measurement noise. As a consequence, the performance criteria (e.g., the sum-of-squared criteria) may differ, but without a reasonable connection to the performance of the point location problem.

In what follows, we aim to discuss the average and the worst-case acceleration of the point location problem shown in the Algorithm 1 when the list of indices is being swapped $\mathcal{I}\leftarrow\widetilde{\mathcal{I}}_i$.

Assumption 3.1. Assume $\mathcal{I}\in\mathbb{N}_{++}^{M}$, $\mathcal{I}=\{1,\dots,M\}$, and let $\widetilde{\mathcal{I}}_i\in\mathbb{N}_{++}^{L_i}$ be evaluated according to Definition 3.1 for $\forall i=\{1,2,\dots,M\}$. Let $M\in\mathbb{N}_{++}$ and $L_i\in\mathbb{N}_{++}$, for $\forall i=\{1,2,\dots,M\}$ be the cardinalities of vectors $\mathcal{I}\in\mathbb{N}_{++}^{M}$ and $\widetilde{\mathcal{I}}_i\in\mathbb{N}_{++}^{L_i}$, respectively. Assume that the following inequality holds

$$M>L_i,\qquad \forall i=\{1,2,\dots,M\}. \tag{8}$$

Although Assumption 3.1 seems to be prohibitive, to the best authors' knowledge, there were no observed such formulations of the relevant CFTOC problems that do not satisfy Assumption 3.1. If (8) does not hold for any $i\in\{1,2,\dots,M\}$, i.e., in the special case when each critical region $\mathcal{R}_j$ is reachable from every critical region $\mathcal{R}_i$ for $\forall i=\{1,2,\dots,M\}$, then the proposed acceleration technique is completely ineffective.

Corollary 3.4.2 (*Worst-case Runtime*)**.** *Given LTI system in (1), CFTOC in (3) leading to the optimal control law in (4), $x(t-T_s)\in\mathcal{R}_i$, and $\widetilde{\mathcal{I}}_i$ according Definition 3.1. Given the full set of indices $\mathcal{I}$ and the M rearranged sets of indices $\widetilde{\mathcal{I}}_i$ according to Definition 3.1. If Assumption 3.1 holds, then replacing $\mathcal{I}$ by $\widetilde{\mathcal{I}}$ in Algorithm 1 reduces the worst-case runtime necessary to evaluate optimal control action $u_0^\star$ for all control steps, except for the initialization of the closed-loop control.*

Proof. If Assumption 3.1 holds, then M and L_i are the cardinalities of $\mathcal{I}$ and $\widetilde{\mathcal{I}}_i$ for $\forall i\in\{1,2,\dots,M\}$, respectively. It follows from Assumption 3.1 that $c>\widetilde{c}_i$ holds for $\forall i\in\{1,2,\dots,M\}$. Therefore, the maximum (i.e., the worst-case) number of iterations that needs Algorithm 1 to execute Step 3 for $i\in\mathcal{I}$ is always greater than for $i\in\widetilde{\mathcal{I}}$. As a consequence, the reduced number of iterations directly reduces the total number of the evaluated flops leading to the reduced corresponding runtime that concludes the proof. □

Remark 3.5 (*Average Runtime*)**.** By Corollary 3.4.2, the average runtime associated with solving the point location problem for $x(t)\in\mathcal{F}$ in Algorithm 1 is also reduced due to the reduced number of explored critical regions $\mathcal{R}_i$ in each control step.

As the consequence of Lemma 3.4 considered for $\widetilde{\mathcal{I}}_i$, for $\forall i=\{1,2,\dots,M\}$, the Algorithm 1 is adopted into the form of Algorithm 3 (Spjøtvold et al., 2006). Note, if Assumption 3.1 holds, the only limitation to reduce the worst-case runtime is the necessity to initialize the first iteration of the point location problem w.r.t. the full explicit partition $\mathcal{F}$, i.e., the so-called warm-start problem of $x(t)\in\mathcal{F}=\cup_{i=1}^{M}\mathcal{R}_i$, see Step 3 in Algorithm 3. On the other hand, each subsequent iteration operates with the reduced set of indices $\widetilde{\mathcal{I}}_{i^\star}$ that contains only $L_i<M$ indices of reachable critical regions.

Algorithm 3 Reachability sets driven sequential search (Spjøtvold et al., 2006).

1: **Inputs:** $x(t)$, list of indices $\mathcal{I}$, M lists of indices $\widetilde{\mathcal{I}}_j$ for $\forall j=\{1,2,\dots,M\}$, feedback law $\kappa(x(t))$
2: **Output:** the optimal control action $u^\star(t)$
3: initialize $\widetilde{\mathcal{I}}_{i^\star}\leftarrow\mathcal{I}$
4: **for** $i\in\widetilde{\mathcal{I}}_{i^\star}$ **do**
5: **if** $x(t)\in\mathcal{R}_i$ **then**
6: update $i^\star\leftarrow i$
7: **break**
8: **end if**
9: **end for**
10: execute $u^\star(t)\leftarrow\kappa(x(t))$ for $i^\star$ per (4)

Remark 3.6. In general, the opened-loop sequence of the optimal control actions computed as the solution of CFTOC in (3) differs from the sequence implemented in the closed-loop control, due to the presence of the terminal constraint in (3d) and/or insufficient length of the prediction horizon N. Nevertheless, the proposed approach is not affected by this discrepancy, as the receding horizon control policy implements a 1-step update that is equivalent for both sequences. As a consequence, a map of reachable critical regions $\widetilde{\mathcal{I}}_i$ according to Definition 3.1 suffice.

3.2. Robustification of reachability analysis

It is well known that the real-time implementation is subjected to a plant-model mismatch and/or bounded external disturbances/perturbations. It is a common practice to address this issue by introducing the additive disturbance into the system model in (1), see Spjøtvold et al. (2006). This results in forming an uncertain LTI system given by:

$$x(t+T_s)=Ax(t)+Bu(t)+Ed(t), \tag{9}$$

where $E\in\mathbb{R}^{n_x\times n_d}$ is disturbance matrix and $d\in D\subset\mathbb{R}^{n_d}$ is the disturbance vector for a given non-empty polytope D containing origin in its strict interior. If the impact of the uncertain parameters is non-negligible, then the *robust* explicit MPC methods should be called, e.g., (Bemporad, Borrelli, & Morari, 2003), etc. Although the robust MPC design introduces some conservativeness into the control law, the industrial implementation may benefit from both, guaranteed performance subject to the disturbances, and reduced complexity of the explicit partition (multiparametric solution), see Ramirez and Camacho (2006). It is a well-known consequence, that the robust explicit MPC design increases the complexity of the construction, as there is an increased number of constraints to be considered, but, on the other hand, the final solution has less critical regions as the consequence of the reduced degree of freedom enforced by considering the impact of disturbances. The design of the uncertainty set D follows the same rules as the robust MPC design, i.e., the controller has robustness guarantees if only if the maximum disturbance is within the considered set D.

In the context of the point location problem, we can include the impact of uncertainness, restricted via the set D, by considering the robust one-step forward a reachable set of $\mathcal{X}$ defined as (Borrelli et al., 2017, Def. 10.17):

$$\text{Reach}(\mathcal{X}, \mathcal{D}) = \{Ax(t) + Bu(t) + Ed(t) \mid \exists x(t) \in \mathcal{X}, \exists u(t) \in \mathcal{U}, \exists d(t) \in \mathcal{D}\}. \quad (10)$$

Thus, $\text{Reach}(\mathcal{X}, \mathcal{D})$ denotes the set of all states that the system (9) can reach within one time step T_s when starting from $\mathcal{X}$, applying any possible control actions from $\mathcal{U}$, and taking into account all possible additive disturbances from $\mathcal{D}$. If all sets $\mathcal{X}$, $\mathcal{U}$, and $\mathcal{D}$ are polytopes, then also the final one-step forward reachable set $\text{Reach}(\mathcal{X})$ is a polytope.

With the same arguments, as we have used to derive the specific formulation of the nominal reachable sets (7), we can also restate (10) as

$$\text{Reach}(\mathcal{X}, \mathcal{D}) = \{(A + BF_{i^\star(t)})x(t) + Ed(t) + Bg_{i^\star(t)} \mid x(t) \in \mathcal{R}_{i^\star(t)}, \exists d(t) \in \mathcal{D}\}. \quad (11)$$

what also boils down to the affine map of the polytope $\mathcal{R}_{i^\star(t)}$ above which the Minkowski sum operation is applied due to the additive disturbances $d(t) \in \mathcal{D}$.

Remark 3.7 (*Parametric Uncertainty*)**.** Note that the resulting one-step reachable set does not need to be always convex, e.g., when a parametric uncertainty is introduced in (9). This problem can be then addressed by a convex approximation, see Kvasnica, Takács, Holaza, and Ingole (2015).

Next, an internal robustification layer of the proposed method is introduced by replacing the original non-robust set of indices $\widetilde{\mathcal{I}}_i$, for $\forall i = \{1, 2, \dots, M\}$ by its robustified alternative $\bar{\mathcal{I}}_i$, for $\forall i = \{1, 2, \dots, M\}$ constructed w.r.t. the robust reachable sets per (10).

Definition 3.2 (*List of Robust Reachable Regions*)**.** Let $\bar{\mathcal{I}}_i \subseteq \mathcal{I} \subset \mathbb{N}^M_{++}$, $\forall i \in \{1, 2 \dots, M\}$ be a list of positive integers corresponding to the indices of the critical regions $\mathcal{R}_j$ robustly reachable from the critical region $\mathcal{R}_i$ as in (10), such that $\text{Reach}(\mathcal{R}_i, \mathcal{D}) \subseteq \cup_{j \in \bar{\mathcal{I}}} \mathcal{R}_j$ holds.

Note, the robustified M sets of indices $\bar{\mathcal{I}}_i$, for $\forall i \in \{1, 2, \dots, M\}$ is evaluated by Algorithm 2 for given $\mathcal{D}$ by replacing the original Step 5 with the following Step:

5 : compute $\text{Reach}(\mathcal{R}_i, \mathcal{D})$ per (11)

Then, the Algorithm 3 is simply adopted to the robustified reachability analysis by replacing the input set of indices $\widetilde{\mathcal{I}}_i \leftarrow \bar{\mathcal{I}}_i$, for $\forall i = \{1, 2, \dots, M\}$ (Spjøtvold et al., 2006).

We would like to point out that even the unexpected disturbance does not lead to the failure of the proposed point location problem acceleration method—because, if the reachability set does not include the current system states then the rest of the partition is systematically explored.

4. Binary encoding of the reachability analysis

In the previous Section 3, we have discussed how to construct integer-valued lists of reachable regions $\widetilde{\mathcal{I}}$ and $\bar{\mathcal{I}}$ per (5) and (10), respectively. In this section, we propose an alternative option for how one can encode the reachability analysis into a compact and memory-efficient binary matrix T. Obviously, this equivalent reformulation leads to the same properties of the original control approach, such as the closed-loop system stability, constraint satisfaction, and recursive feasibility.

Consider a square binary matrix $T \in \{0, 1\}^{M \times M}$ of a fixed size M, where its columns denote critical regions $\mathcal{R}_j$ for $j \in \{1, \dots, M\}$ that are reachable from the critical region $\mathcal{R}_i$. Analogously, $i \in \{1, \dots, M\}$ denotes the row of the matrix T. The construction procedure of this binary reachability matrix T is summarized in Algorithm 4.

Information if the intersection $\text{Reach}(\mathcal{R}_i) \cap \mathcal{R}_j$ is (not) an empty set is stored in the binary matrix T such that $T_{i,j} \in \{0, 1\}$, see Step 8 in the Algorithm 4. To be concrete, $T_{i,j} = 1$ denotes that the critical region $\mathcal{R}_j$ is reachable from $\mathcal{R}_i$ within one sample step T_s, under the map (1), and

Algorithm 4 Binary encoded reachability analysis.

1: **Inputs:** polyhedral partition $\mathcal{P} = \cup_{i=1}^{M} \mathcal{R}_i$ of $\kappa(\cdot)$, list of indices $\mathcal{I}$
2: **Output:** list of reachable critical regions $T \in \{0, 1\}^{M \times M}$
3: define $T \leftarrow \mathbf{0}_{M \times M}$
4: **for** $i \in \mathcal{I}$ **do**
5: compute $\text{Reach}(\mathcal{R}_i)$ per (5)
6: **for** $j \in \{1, \dots, M\}$ **do**
7: **if** $\text{Reach}(\mathcal{R}_i) \cap \mathcal{R}_j \neq \emptyset$ **then**
8: $T_{i,j} \leftarrow 1$
9: **end if**
10: **end for**
11: **end for**

by using a control action from $\mathcal{U}$. And the other way around, $T_{i,j} = 0$ denotes that $\mathcal{R}_j$ is not reachable from $\mathcal{R}_i$ per (5). Finally, by iterating through all critical regions $\mathcal{R}_i$ with $i \in \{1, \dots, M\}$ we construct the binary matrix $T \in \{0, 1\}^{M \times M}$ that concludes our reachability analysis.

Implementation of T into the Algorithm 3 is straightforward. At the beginning of each iteration, i.e., in Step 4 of Algorithm 3, we need to substitute $\widetilde{\mathcal{I}}_i$ by the $i^\star$-th row of T as follows

$$\widetilde{\mathcal{I}}_{i^\star} \leftarrow T_{i^\star} \quad \text{if} \quad x(t - T_s) \in \mathcal{R}_{i^\star}, \quad (12)$$

where $i^\star$ is the index of critical region $\mathcal{R}_{i^\star}$ active at the previous time instance $t - T_s$.

From the real-time evaluation point of view, $T_{i^\star}$ is a binary vector that has to be translated into an appropriate format or additional bit-shifting functions need to be used. On the other hand, implementation of $\widetilde{\mathcal{I}}_i$ (or $\bar{\mathcal{I}}_i$) requires no additional manipulations or functions as it directly uses the list of indices to point all reachable critical regions, see Step 4 in Algorithm 3.

Generally, compared to the reachability matrix T, the list of indices $\widetilde{\mathcal{I}}_i$ needs to store much fewer variables, i.e., vector of indices of reachable critical regions. The problem is that $\widetilde{\mathcal{I}}$ is encoded as an integer-valued vector the size of which is dictated by the complexity of explicit MPC, i.e., the number of constructed critical regions M. By assuming α-bit unsigned integer format and if $M < (2^\alpha - 1)$ holds, then the worst-case memory footprint of $\widetilde{\mathcal{I}}_i$ is (αM^2)-bits. On the other hand, the binary values occupy significantly less memory compared to the storage of the vectors of integers. The memory footprint of the binary matrix T is hence only M^2-bits.[2] We recall, that the number of constructed critical regions M can be reduced by some well-known simplification technique, e.g., optimal region merging, clipping, etc., see Kvasnica, Holaza, Takács, and Ingole (2015). Needless to say, the same memory comparison also holds for the list $\bar{\mathcal{I}}_i$ that is encoded as $\widetilde{\mathcal{I}}_i$.

Remark 4.1 (*Conversion of Lists*)**.** We have that T contains the same information as $\widetilde{\mathcal{I}}_i$, $\forall i \in \{1, 2, \dots, M\}$. The only difference is in the form of how these reachability analyses are encoded. It can be shown that $\widetilde{\mathcal{I}}_i$ can be easily transformed into T and vice versa. The same applies to the robustified list of indices $\bar{\mathcal{I}}_i$ constructed for the uncertain LTI system in (9).

Remark 4.2 (*Implementation Redundancy*)**.** Notice that the robust reachable set (10) considers bounded external disturbances/perturbations from $\mathcal{D}$, where these bounds are given as a tuning parameter. If an unexpected disturbance $d(t) \notin \mathcal{D}$ is introduced to the system then the modified point location Algorithm 1, i.e., being updated by (12), does not guarantee to compute the $u^\star(t)$. Even though it is an issue of an

[2] It is assumed that we have access to each individual bit. If this is not true, then we need to assume a rounded-up multiplication of the smallest available data format.

incorrectly selected $\mathcal{D}$, the Algorithm 1 can be forced to find the $u^\star(t)$. The solution is twofold. If the binary matrix T is used, then we can set complementary values of the original binary vector,i.e., $\mathcal{I} \leftarrow \mathbf{1}_M - T_{i^\star}$, and restart the Algorithm 1. We recall that the $\mathbf{1}_M$ denotes the vector of ones of size M. On the other hand, if $\widetilde{\mathcal{I}}_i$ is used, we need to call a set difference $\mathcal{I} \leftarrow \{1,\dots,M\}/\widetilde{\mathcal{I}}_{i^\star}$ and restart the Algorithm 1. The same statement holds also for the list $\bar{\mathcal{I}}_i$.

5. Acceleration of the real-time evaluation using the sorted list

In previous sections, we have shown that one can use reachability analysis to create lists of indices $\widetilde{\mathcal{I}}_i$, or $\bar{\mathcal{I}}_i$, to accelerate the point location problem via decreasing and specifying regions that need to be explored. Moreover, it was proposed that without loss of generality, we can encode these lists into the memory-efficient binary-valued matrices T. Next, in this section, we aim to propose an additional layer, which can further make the real-time evaluation of explicit MPC policies *faster*. Specifically, as shown in Holaza et al. (2020), the online evaluation of optimal control actions can be accelerated with an appropriate ordering of indices. Even though this method is also fully compatible with our approach, we suggest an effective alternative methodology to sort indices in each list of $\widetilde{\mathcal{I}}_i$, or $\bar{\mathcal{I}}_i$, for $\forall i = \{1,2,\dots,M\}$, which is easily adoptable by all of our reachability analysis algorithms. We point out that introducing the evaluation of the sorted list preserved the properties of the original control approach, such as the closed-loop system stability, constraint satisfaction, and recursive feasibility. The main benefit of the proposed acceleration layer is that this method does not introduce any real-time burden into the evaluation of optimal control action, as the sorting procedure is evaluated fully offline.

Assume a function $V_{i,j} : \mathcal{X} \mapsto \mathbb{R}$ mapping a set of states $\mathcal{X}$ into a scalar value. Technically, $V_{i,j}$ takes the intersection between the ith reachable set and the jth critical region of $\mathcal{F}$ and associate this polytope with its volume, i.e., we have that

$$V_{i,j} = \text{Volume}(\text{Reach}(\mathcal{R}_i) \cap \mathcal{R}_j). \tag{13}$$

We aim to sort indices of each list $\widetilde{\mathcal{I}}_i$ such that

$$\hat{\mathcal{I}}_i = \{\hat{\mathcal{I}}_{i,1}, \hat{\mathcal{I}}_{i,2}, \dots, \hat{\mathcal{I}}_{i,L_i}\}, \tag{14}$$

where $L_i \leq M$ is the number of reachable critical regions[3] from $\mathcal{R}_i$ and $V_{i,l} \geq V_{i,l+1}$, $\forall l \in \{1,2,\dots,L_i\}$ holds. The rationale behind this sorting is that the probability of $x(t) \in \mathcal{R}_i$ being mapped into $x(t+T_s) \in \mathcal{R}_j$ grows proportionately with their intersection volume $V_{i,j}$. Therefore, the vector $\hat{\mathcal{I}}_i$ is ordered in descending order to the volumes $V_{i,j}$.

Note, the same approach holds for the robustified counterpart $\bar{\mathcal{I}}$ according to Definition 3.2.

Remark 5.1 (*Sorting of the Binary Indices*)**.** Sorting of indices can be applied also to the binary matrix T via employing permutation matrices. The cost of this acceleration is, however, in increased memory requirements. As each row of T would need one permutation matrix, the final memory footprint would raise from M^2-bits to (M^2+M^3)-bits.

6. Numerical analysis

First, we demonstrate in detail the main benefits of the proposed acceleration techniques using the illustrative numerical example. Next, we implement this method on the embedded platforms and validate the control strategy using the challenging unstable, and higher-dimensional benchmark systems with fast dynamics—the ball on-plate device and the inverted pendulum device.

6.1. Illustrative example

To illustrate in detail the proposed acceleration approach from Section 3, together with its efficient binary encoding from Section 4, and sorted list according Section 5, we investigate the well-known benchmark LTI system as in (1) with matrices

$$A = \begin{bmatrix} 0.5403 & -0.8415 \\ 0.8415 & 0.5403 \end{bmatrix}, \quad B = \begin{bmatrix} -0.4597 \\ 0.8415 \end{bmatrix}, \tag{15}$$

that represents an oscillatory system ball on a rope, where the state vector x denotes the position and speed of the ball, respectively, and the input action u is force added to the ball. We formulated the CFTOC optimization problem as in (3) with prediction model (3b) given in (15). State and input constraints were set to $|x| \preceq [10, \infty]^\top$ and $|u| \preceq 1$, respectively. Weighting matrices were chosen as $Q_x = I$ and $Q_u = 1$, where I is the identity matrix of appropriate dimensions. The prediction horizon was set to $N = 3$ and both terminal set $\mathcal{X}_N$ in (3d) and terminal penalty $\|x_k\|^2_{Q_x}$ were omitted. By solving the CFTOC problem in (3) parametrically, via the `MPT3` toolbox (Herceg, Kvasnica, Jones, & Morari, 2013), we obtained explicit feedback law $\kappa(x(t))$ as in (4) defined over $M = 13$ regions in 4.95 seconds.[4] The corresponding polyhedral partition $\mathcal{F}$ is shown in Fig. 1, where the index of each critical region is depicted per unsorted list $\mathcal{I}$. The closed-loop evaluation of this controller can be carried out by using the sequential search method described in the Algorithm 1. In what follows, we aim to accelerate this algorithm by replacing $\mathcal{I}$ with a new list of indices.

Firstly, we have constructed the list of reachable critical regions $\widetilde{\mathcal{I}}$ per Algorithm 2, which took 1.46 s. We recall that $\widetilde{\mathcal{I}}$ denotes the list of $i = 1,\dots,M$ vectors where each of this vector $\widetilde{\mathcal{I}}_i$ contains indices of all critical regions that are reachable when starting from the ith region $\mathcal{R}_i$ and under the map (1) with matrices (15). To illustrate this list, let us select the $\widetilde{\mathcal{I}}_2$ that has the form of sorted sequence of integers:

$$\widetilde{\mathcal{I}}_2 = \{2,4,5,9,10,11\}, \tag{16}$$

and the corresponding 2-nd row of the binary matrix T has the form of:

$$T_2 = [0,1,0,1,1,0,0,0,1,1,1,0,0], \tag{17}$$

i.e., having the non-zero entries at the corresponding positions of reachable regions. Based on this result, we declare that if the state vector starts within the second critical region, i.e., $x(t) \in \mathcal{R}_2$, denoted as the blue polytope in Fig. 1, then the controlled system in (15) with the explicit feedback law $\kappa(x(t))$ will map the consecutive state vector $x(t+T_s)$ onto the reachable set $\text{Reach}(\mathcal{R}_2)$, depicted as the dashed polytope in Fig. 1. As all non-empty intersections of $\text{Reach}(\mathcal{R}_2)$ with $\mathcal{F}$ are known, we have that only regions with indices in (16) (or in (17)) have to be considered for allocation of the optimal control action $\kappa(x(t+T_s))$. Hence, for the discrete time $(t+T_s)$, we call $\mathcal{I} \leftarrow \widetilde{\mathcal{I}}_2$ in the Step 4 of the Algorithm 3. This minimizes the total number of explored critical regions from $M = 13$ to 6 which decreases the online execution time of the $\kappa(x(t+T_s))$.

Moreover, we have analyzed the further acceleration layer presented in Section 5. As can be seen in Figure 1, the intersection of $\text{Reach}(\mathcal{R}_2)$ with the feasible partition $\mathcal{F}$ yields polytopes of different volumes. Each non-empty intersection is shown in Fig. 1 with a dedicated color. By computing volumes of these intersections per (13) we can easily sort indices in (16) leading us to the reordered list (14) of form

$$\hat{\mathcal{I}}_2 = \{10,5,4,11,2,9\}. \tag{18}$$

The main motivation to use the sorted list (18), instead of the original list in (16), is that generally there are more states $x(t+T_s) \in \mathcal{R}_{10}$ than in any other critical region listed in (16). Thus, the highest change that the Algorithm 1 will determine $\kappa(x(t+T_s))$ is exactly by exploring $\mathcal{R}_{10}$.

[3] The number of reachable critical regions is denoted by the number of non-empty interactions $\text{Reach}(\mathcal{R}_i) \cap \mathcal{R}_j \neq \emptyset$, $\forall j = 1,\dots,M$.

[4] AMD Ryzen 7 PRO 5850U with 16 GB DDR4 RAM and running Windows 11 Pro, MATLAB R2022b, YALMIP R20210331, MPT v3.2.1.

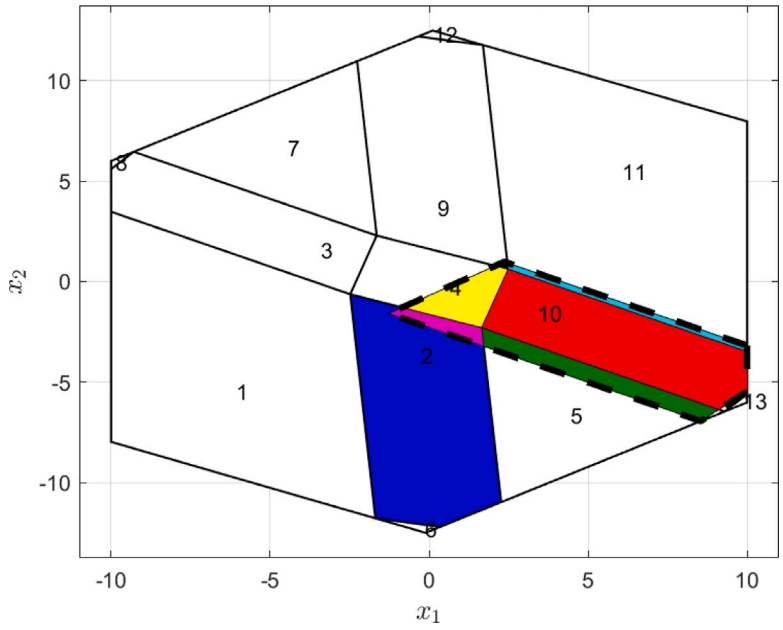

Fig. 1. Polytopic partition of CFTOC problem from Section 6.1. All critical regions are indexed by the original list $\mathcal{I}$. The dashed critical region denotes the reachable set computed by (5) when considering the set of initial states given by the blue critical region $\mathcal{R}_2$. All non-empty intersections of the reachable set with each critical region of the partition $\mathcal{P}$, performed at the fifth step of the Algorithm (2), are shown with a unique color.

With the same argument, this is the reason why the list in (18) is sorted in ascending order.

To determine the efficiency of the proposed method, we have performed numerical simulations. Specifically, the closed-loop was composed of the designed explicit MPC $\kappa(\cdot)$ and the model in (15). We have chosen M initial conditions as random state vectors from each critical region of the partition $\mathcal{P}$. The number of simulation steps was set to $N_{\text{sim}} = 30$. Three test cases were considered that differed only in the applied list of indices in the point location Algorithm 1. In the first case, we used the original list of indices $\mathcal{I}$. In the second one, we have exploited the binary encoded list T, generated by the Algorithm 4. Finally, in the last third case, we employed the same strategy as in the previous case with the exception that the sorted list of indices $\hat{\mathcal{I}}$ was used. We recall that the first simulation step of the Algorithm 1 always uses $\mathcal{I}$ to determine the active index $i^\star$. Afterwards, this list is substituted by $T_{i^\star}$, or $\hat{\mathcal{I}}_{i^\star}$, respectively.

We point out that the sequential search Algorithm 1 traverses through regions to determine the active index $i^\star$ where $x(t) \in \mathcal{R}_i^\star$. Each region of $\mathcal{P}$ is defined as a polytope that is represented by a finite number of intersected half-spaces. Hence, to determine $i^\star$ we need to explore a finite number of half-spaces, of a fixed size. This provides us with a reasonable indicator of computational resources that are required to evaluate the optimal control law of $\kappa(x(t))$. Therefore, as the performance indicator we have chosen the minimal, average, and maximal number of explored half-spaces needed to determine the control law $\kappa(x(t))$ for $t = 1, \ldots, \bar{N}_{\text{sim}}$ and for all $M = 13$ initial conditions. For the sake of a fair comparison, we have considered only simulation steps $\bar{N}_{\text{sim}} \leq N_{\text{sim}}$ that were required to converge the "terminal region", i.e., the critical region that contains the origin.[5]

The results are shown in Table 1. Here, for ease of presentation, we have considered simulation results with the original list $\mathcal{I}$ as the reference, and for all other utilized lists we show the respective ratio, i.e., the number of explored half-spaces obtained with $\mathcal{I}$ divided by the associated number needed with T or $\hat{\mathcal{I}}$, respectively. In other words, the number 1.00 represents the reference value where no acceleration/deceleration was achieved. Any larger number than 1 represents a proportional acceleration, and any smaller number than 1 denotes the deceleration.[6] The Table 1 shows that by using the binary encoded reachability list of indices T, we needed to evaluate only 1 013 half-spaces, compared to the 1 758 required for the original list of indices $\mathcal{I}$, what represents improved efficiency by a factor of 1.73. By using the sorted reachability list of indices $\hat{\mathcal{I}}$ we have accelerated the point location problem even more to 2.09 on average. It should be noted that these improvements were achieved w.r.t. extended memory requirements needed to store index lists[7] T or $\hat{\mathcal{I}}$. In our case, the memory footprint of the receding horizon control law of $\kappa(\cdot)$ was 1.48 kB, and the memory requirement for the binary list T was 0.1 kB, and for the ordered real-valued list $\hat{\mathcal{I}}$ equal to 0.29 kB. This represents a memory increase of 6.8 % and 19.6 %, respectively. To conclude, the real-time evaluation burden was accelerated on average up to 109% at a cost of increased memory demands ≈ 20 %.

Table 1
Acceleration factors of explored half-spaces (using the reachability analysis per (5)).

List of indices	min	avg	max
$\mathcal{I}$ (default)	1.00	1.00	1.00
T (binary, Section 4)	1.48	1.73	1.94
$\hat{\mathcal{I}}$ (sorted, Section 5)	1.56	2.09	2.83

For convenience, the same case study was performed also considering the robust reachability analysis per (10), where the bounded disturbance of $|d(t)| \leq 0.01$ was assumed. The computation of this list took 1.61 s. We obtained the list of indices $\bar{\mathcal{I}}$, where the second row of this list was defined as

$$\bar{\mathcal{I}}_2 = \{2, 4, 5, 10, 13\} \tag{19}$$

[5] The reason for this setup was that the parametric solver, used to compute the explicit solution (4) does not necessarily set the critical region as the first position of the index list $\mathcal{I}$. As our proposed method tackles this issue, it would greatly benefit if we kept the $\bar{N}_{\text{sim}} = N_{\text{sim}}$.

[6] In another word, the factor of, e.g., 1.56 represents a 56 % acceleration as 1.56-times less operations were needed to compute the optimal control action.

[7] For simplicity, we are omitting discussions on additional operations needed to carry out the implementation of the binary encoded T, i.e., the transformation of a binary vector to a real-valued one.

Table 2
Acceleration factors of explored half-spaces (using the reachability analysis per (10)).

List of indices	min	avg	max
$\mathcal{I}$ (default)	1.00	1.00	1.00
T (binary, Section 4)	1.30	1.82	2.23
$\hat{\mathcal{I}}$ (sorted, Section 5)	1.21	2.03	2.83

and its sorted version as

$$\hat{\mathcal{I}}_2 = \{10, 5, 2, 4, 13\}. \tag{20}$$

By comparing the robustified list (19) with its respective counterpart (16) we can observe that a different set of regions is now reachable due to the introduced disturbances. Moreover, notice that the sorted set (20) altered the order of indices that are contained also in (18) due to the changed volumes of respective intersections (13). The same simulation analysis was performed as in the previous nominal case. The results are shown in Table 2, where we can notice relatively similar accelerations as depicted in Table 1. Finally, the memory footprint of $\bar{\mathcal{I}}$ was 0.21 kB and its binary encoded version $\hat{\mathcal{I}}$ was 0.10 kB.

6.2. Demonstrative 2D example

To analyze the acceleration factors using the implementation on the embedded hardware platforms, we consider a well-known benchmark of ball on beam unstable system with fast-dynamic.[8] The linearized discrete-time LTI system in (1) in the discrete-time domain using the sampling time 10^{-3} s has the following matrices:

$$A = \begin{bmatrix} 1 & 0.01 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} -0.4 \\ -70.1 \end{bmatrix} \times 10^{-3}, \tag{21}$$

where two system states represent the position and the speed of the ball on the plate in its x-axis coordinate. The control input represents the force added to the servos adjusting the tilt angle of the beam. We assume that two system states are subjected to the physical constraints: $-0.2 \leq x_1 \leq 0.01$, $-0.1 \leq x_2 \leq 0.1$, while the control action is under the symmetric physical constraints: $|u| \leq 0.0524$.

The explicit MPC design problem in (3) was designed for weighting matrices in (3a) were selected as $Q_x = \mathrm{diag}(100, 10)$, $Q_u = 1$. The terminal penalty Q_N was chosen as the solution of the discrete-time Riccati equation and the terminal set $\mathcal{X}_N$ was omitted.[9] The length of the prediction horizon was set to $N = 20$ samples. By utilizing the MPT3 toolbox (Herceg, Kvasnica, Jones, & Morari, 2013), we have obtained a parametric solution as in (4) in 16.30 s and the associated memory footprint was 51.32 kB. The resulting feedback law $\kappa(\cdot)$ was defined over $M = 426$ critical regions that were ordered by a *default* list $\mathcal{I}$.

The runtimes of both, the conventional explicit MPC controller (Algorithm 1) and the proposed reachability-sets-driven explicit MPC controller (Algorithm 3) with the sorted listed constructed according to Section 5, were analyzed, respectively. We note that the sorted list $\hat{\mathcal{I}}$ was constructed in 45.83 s and its memory footprint was 4.29 kB, i.e., an increase of 8.36 % of the controller's memory.[10]

The *acceleration level* was determined based on the numerical simulations, where the discretized version of (22) was controlled by the explicit MPC policy $\kappa(\cdot)$. As a representative set of initial conditions, we have used a set of 50 randomly generated points, i.e., 50 feasible initial conditions x_0 such that x_0 are not included in the terminal (unconstrained) set. We recall that, to provide a fair comparison, the acceleration performance is considered only until the terminal regions are reached.

Table 3
Runtimes and acceleration factors evaluated for ball on beam benchmark implemented on the embedded platform ESP32 in 10^{-6} s.

List of indices	min	avg	max
$\mathcal{I}$ (default)	68	563	3 432
$\hat{\mathcal{I}}$ (sorted, Section 5)	57	86	188
Acceleration factor	1.19	6.54	18.26

Table 4
Runtimes and acceleration factors evaluated for ball on beam benchmark implemented on the embedded platform ESP32-S3 in 10^{-6} s.

List of indices	min	avg	max
$\mathcal{I}$ (default)	57	456	2 918
$\hat{\mathcal{I}}$ (sorted, Section 5)	52	70	132
Acceleration factor	1.10	6.51	22.11

To make the runtime results fully comparable, both investigated algorithms were implemented in the C code. Then, the proposed acceleration method was implemented on the embedded hardware to analyze the real-time runtimes. Specifically, the explicit MPC controllers were implemented on two embedded hardware platforms: (i) the widely-used ESP32 DevKit V4 equipped with a 32-bit microprocessor with a clock frequency of 240 MHz and 4 MB of flash memory and (ii) the new generation ESP32-S3 equipped with a Dual 32 bit Xtensa LX7 cores running up to 240 MHz and 16 MB of the flash memory.

First, the measured runtimes for ESP32 are summarized in Table 3, where minimum, average, and maximum runtimes are evaluated in microseconds, respectively. Table 3 shows also the computed acceleration factors for each of the analyzed cases. As can be seen, the proposed acceleration method significantly reduced the real-time runtimes on the embedded platform, as the runtime decreased by factor 6 on average. The minimum runtimes (best-case), evaluated as the mean values of the minimum runtimes of 50 closed-loop trajectories, are also minimized. Moreover, the worst-case runtimes, evaluated as the mean values of the maximum runtimes of 50 closed-loop trajectories, were decreased by the order of magnitude.

Next, the measured runtimes for ESP32-S3 are summarized in Table 4 for the same set of 50 initial conditions to make the results in Table 3 comparable. In Table 4, the minimum, average, and maximum runtimes are again evaluated in microseconds. Table 4 shows also the evaluated acceleration factors for each of the analyzed cases. Analogous to the results presented in Table 3, the proposed acceleration method significantly reduced the real-time runtimes on the new generation embedded platform, as the runtime also decreased by factor 6 on average. As expected, the absolute values of runtimes are lower when implementing the control algorithms on the new generation platform ESP32-S3, while the relative values of the acceleration factors of the proposed method remain preserved, cf. Table 4. To conclude, as can be observed from data in Tables 3, 4, both, the minimum runtimes and the worst-case runtimes, were significantly reduced.

6.3. Demonstrative 4D example

Finally, let us consider a four-dimensional fast-dynamic system of an inverted pendulum on a cart (Bakaráč, Valiauga, & Kvasnica, 2018).

[8] Benchmark data were adopted from: https://github.com/ferreau/mpcBenchmarking.

[9] The terminal set was not included in the optimization problem, due to the dynamics of (22) it converged only to a single point — origin.

[10] For comparison, using MPT3, we have also built a memory-optimized binary search tree of depth 12 and 1 072 nodes. Its construction took 978.47 s and its memory footprint was 41.88 kB. Hence, for this specific case, the binary search tree has a more favorable memory demand as it requires 24.71 % less storage than our proposed approach.

Linearizing dynamics of this system around its upright stable position yields the following LTI model:

$$\begin{bmatrix} \dot{\rho} \\ \ddot{\rho} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 36.77 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \rho \\ \dot{\rho} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 3.75 \\ 0 \\ 1 \end{bmatrix} u, \tag{22}$$

where ρ, $\dot{\rho}$, ϕ, and $\dot{\phi}$ denotes the position of the cart, the cart's velocity, the pendulum's angle from the upright position, and the angular velocity, respectively. The control input u represents the force added to the card. We assume that these states are subjected to the physical constraints $|\rho| \leq \pi/12$, $|\dot{\rho}| \leq 20$, $|\phi| \leq 0.25$, and $|\dot{\phi}| \leq 2$, while the control action is constricted by $|u| \leq 10$. The system in (22) was discretized with sampling period $T_s = 0.08$ seconds and used as the prediction model in (3b). The MPC optimization problem in (3) was designed with squared Euler's norm $p = 2$, length of the prediction horizon was set to $N = 3$ samples, and weighting matrices in (3a) were selected as $Q_x = \mathrm{diag}(10^4, 10^0, 10^2, 10^0)$, $Q_u = 10^{-3}$. The terminal penalty Q_N was chosen as the solution of the discrete-time Riccati equation and the terminal set $\mathcal{X}_N$ was omitted.[11] The final CFTOC optimization problem had the form of a QP with four states and three optimized variables. By utilizing the MPT3 toolbox (Herceg, Kvasnica, Jones, & Morari, 2013), we have obtained a parametric solution as (4) in 15.11 s, where the feedback law $\kappa(\cdot)$ was defined over $M = 229$ critical regions that were ordered by a *default* list $\mathcal{I}$.

The reachability analysis was performed as per Algorithm 2 by considering robust reachable sets (10) with $|d(t)| \leq 0.01$. The constructed list of reachable regions $\widetilde{\mathcal{I}}$ was then sorted as described in Section 5 yielding $\hat{\mathcal{I}}$, and then converted into the binary matrix as per Section 4 returning the set (binary matrix) T. From the computation point of view, construction of the list $\widetilde{\mathcal{I}}$ took 830.70 s, the sorting 0.01 s, and the conversion into $\hat{\mathcal{I}}$ consumed 0.06 s of the offline time. These two sets were then used to accelerate the sequential search point location problem of real-time evaluation of the optimal control action.

The *acceleration level* was determined based on the numerical simulations, where the discretized version of (22) was controlled by the explicit MPC policy $\kappa(\cdot)$. As a set of initial conditions, we have used a random point from each region of $\kappa(\cdot)$, i.e., 229 initial conditions. The number of simulation steps was set to 10^3 which was sufficient to converge to the terminal region, i.e., the critical region containing the origin. We remind you that, to provide a fair comparison, the acceleration performance is considered only until the terminal regions are reached. Moreover, we disregard the computation needed for conversion from the binary encoding when using the list T.

The generated results are shown in Table 5. Here we can observe that the sequential search, exploiting the binary reachable list T, was accelerated on average by a factor of 2.60, while the minimal acceleration was 1.00 and the maximal one was 3.27. On the other hand, by using the sorted list $\hat{\mathcal{I}}$, we recognized even further acceleration with average, minimal, and maximal factors of 0.81, 8.93, and 19.19, respectively. Notice that the minimal acceleration is lower than the reference value 1 which implies that our proposed method has decelerated computation of the point location problem. This phenomenon occurred only a few times due to the performed region sorting and it is dependent on selected initial conditions. Notice that when the binary (unsorted) list T was used, no deceleration was detected. Moreover, the additional memory footprint of binary reachable lists T was 8.95 kB, and for the $\hat{\mathcal{I}}$ it was 1.79 kB. This represents an increase of 10 % and 2 %, respectively, w.r.t. the memory requirement of the EMPC policy, which was determined at 90.55 kB.

The main novelty w.r.t. Spjøtvold et al. (2006) can be demonstrated by results shown in the Table 5. Firstly, the unsorted reachability list T has the same[12] acceleration performance as the list suggested by Spjøtvold et al. (2006). However, its binary encoding allows us to decrease the memory footprint of the list by 80 %, i.e., from 8.95 kB to 1.79 kB. Secondly, only by sorting the reachability list, the online acceleration can be increased by 343 %, i.e., from the factor of 2.6 to 8.93 on average. Needless to say, these improvements were achieved without losing any control properties of the original MPC problem (3), and only at the expense of minor additional offline computation resources, see (13).[13]

Table 5
Acceleration factors of explored half-spaces for inverted pendulum benchmark.

List of indices	min	avg	max
$\mathcal{I}$ (default)	1.00	1.00	1.00
T (binary, Section 4)	1.00	2.60	3.27
$\hat{\mathcal{I}}$ (sorted, Section 5)	0.81	8.93	19.19

Table 6
Runtimes and acceleration factors evaluated for inverted pendulum benchmark implemented on the embedded platform ESP32 in 10^{-6} s.

List of indices	min	avg	max
$\mathcal{I}$ (default)	55	1 896	4 213
$\hat{\mathcal{I}}$ (sorted, Section 5)	65	126	482
Acceleration factor	0.9	15.1	8.7

Finally, the proposed acceleration method was implemented on the embedded hardware to investigate the real-time runtimes. The runtimes of both, the conventional explicit MPC controller (Algorithm 1) and the proposed reachability-sets-driven explicit MPC controller (Algorithm 3) with the sorted listed constructed according to Section 5, were analyzed, respectively. Analogous to the previous demonstrative case study in Section 6.2, to make the results fully comparable, both algorithms were implemented in the C code and ran on the microcontroller platform ESP32 DevKit V4 equipped with a 32-bit microprocessor with a clock frequency of 240 MHz and 4 MB of flash memory. The illustrative simulation of the closed-loop control was evaluated for an initial condition $x_0 = [0.1, 0, 0.2, 0]^\top$ in (3e). The measured runtimes are summarized in Table 6, where minimum, average, and maximum runtimes are evaluated in microseconds, respectively. Table 6 shows also the computed acceleration factors for each of the analyzed cases. As can be seen, the proposed acceleration method significantly reduced the real-time runtimes on the embedded platform, as the runtime decreased by a factor of 15 on average. Moreover, also worst-case runtime was decreased by the order of a magnitude.

To summarize the generated results, data from all three case studies are put together and presented in Table 7. In this table, we can observe the scaling behavior of our proposed method and the binary search approach concerning the number of critical regions (M) and the domain dimension. For better clarity, we note that each number is stored as a double precision variable requiring eight bytes of memory per number. For instance, the footprint of a binary search tree was calculated using a well-known approach as $n_{\text{node}}(n_x + 1 + 2) \times 8$, where for each node n_{node} we stored the separating hyperplane of dimension $(n_x + 1)$, and 2 indices pointing to nodes in the subsequent level of the tree. For our proposed method, the memory footprint encompasses requirements for both the original explicit MPC and the constructed reachable list. From the Table 7, we can observe that both the offline computation time ("time") and memory requirements

[11] The terminal set was again not included in the optimization problem, due to the dynamics of (22) it converged only to a single point — origin.

[12] If we omit the transformation of a binary vector into a real-valued one.

[13] For comparison, the memory-optimized binary search tree from MPT3 toolbox was created in 583.85 s. The tree consisted of 8 076 nodes, had a depth of 19, and its memory footprint was 441.65 kB which represents more than four times memory increase compared to our proposed approach.

Table 7
Memory and offline computation comparison.

	Explicit MPC			Binary tree		Proposed method	
	M	Time [s]	Size [kB]	Time [s]	Size [kB]	Time [s]	Size [kB]
Illustrative example (Section 6.1)	13	4.95	1.48	1.15	0.82	1.61	1.77
Demonstrative 2D example (Section 6.2)	426	16.30	51.32	978.47	41.88	45.83	55.61
Demonstrative 4D example (Section 6.3)	229	15.11	90.55	583.85	441.65	830.7	92.34

("size") of the reachable list scale almost linearly with the number of regions. However, as the dimensionality of the problem increases, the offline computation grows non-linearly. Regarding the binary search technique, the increased number of regions significantly impacts computation time, while greater dimensional problems can lead to an unfavorable tree structure with numerous sub-regions, resulting in high memory consumption. It is essential to emphasize that the results of these two methods are highly sensitive to specific cases. Therefore, we encourage readers to investigate multiple point location approaches and select the one that best suits the targeted process.

Remark 6.1 (*Possible Numerical Issues*)**.** It is known that constructing reachable sets is a computationally expensive and numerically sensitive operation. Therefore, when dealing with higher-dimensional systems one can encounter numerical issues that will inhibit computation of the desired reachable set(s). The straightforward way how to approach this issue is to simply skip the computation of this set leading to a "gap" in the reachable list $\widetilde{\mathcal{I}}$. This does not represent an issue, see Remark 4.2, as it will be only at a price of providing no acceleration for the given region, i.e., the standard sequential search algorithm would be used. However, a more efficient approach would be to compute an approximated reachable set.

7. Conclusions

This paper presented a novel acceleration approach for the online evaluation of explicit MPC policies. It was shown that while the conventional framework of the real-time sequential search considers traversing through all the critical regions, the number of explored regions can be mitigated by altering the list of exploring regions. Specifically, we directly extend the results introduced in Spjøtvold et al. (2006) to show how to a-priory exploit the reachability analysis to create compact yet sufficient lists of indices that can be then used during the online evaluation procedure by carefully choosing the list respective to the current state of the controlled system. It was discussed that such staged online Algorithm 3 preserves both recursive satisfaction of the original constraints as well as the asymptotic stability, i.e., if the MPC optimization problem guaranteed these properties in the first place.

Two types of reachability analysis were considered. The first one assumed only the nominal model (5) to create the reachable sets, while the second one took into account bounded additive disturbances (10). Both approaches construct lists of indices, generally with different memory footprints, that need to be stored on a targeted control platform to enable acceleration of the online evaluation of explicit feedback laws. To address this drawback, we have proposed a memory-efficient binary encoding. The implementation details, benefits, and limitations of the binary encoding and its alternative integer-vector-based lists were discussed in detail.

Further acceleration method, gaining the benefits of the reachability analysis introduced in Spjøtvold et al. (2006), was proposed. The volumes of the critical regions intersecting the reachability sets were evaluated and sorted to maximize the probability of accelerating the real-time evaluation of the point location problem. We point out, that this acceleration method does not introduce any real-time burden. Although the worst-case boundary on the evaluation time was not reduced, it has been demonstrated that the worst-case runtimes are reduced for all control steps except for the initial one. As a consequence, the average runtimes are significantly reduced.

The efficiency of the proposed acceleration method was demonstrated via extensive case studies. It was shown that, at the cost of the increased memory footprint, the average point location runtimes could be significantly reduced. In the case of the considered illustrative numerical example, the memory demands were increased by about 20 %, and the computational complexity decreased up to 109 % on average. In the case of the validation using the two-inputs and two-outputs fast-dynamic system of the ball on beam benchmark, the acceleration factor on the embedded platforms reach the level of about 7-times real-time speed-up on average. Finally, in the case of the validation using the four-dimensional fast-dynamic system of the inverted pendulum benchmark, the acceleration factor on the embedded platform reaches the level of about 9-times real-time speed-up on average. The price for this improved online evaluation was equal to the 10% of the original explicit MPC policy.

The next research is focused on the explicit extension of the binary reachability matrix towards the binary search trees, as the binary matrix T could be straightforwardly transformed into the decision rules indicating the branches/nodes to be explored (or pruned/skipped).

CRediT authorship contribution statement

Juraj Holaza: Conceptualization, Data curation, Investigation, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **Peter Bakaráč:** Data curation, Investigation, Software, Validation, Writing – review & editing. **Juraj Oravec:** Conceptualization, Formal analysis, Funding acquisition, Investigation, Project administration, Supervision, Validation, Writing – original draft, Writing – review & editing.

Acknowledgments

The authors gratefully acknowledge the contribution of the Scientific Grant Agency of the Slovak Republic under the grants 1/0490/23, 1/0297/22, and the Slovak Research and Development Agency under the project APVV-20-0261. This research is funded by the European Union's Horizon Europe under grant no. 101079342 (Fostering Opportunities Towards Slovak Excellence in Advanced Control for Smart Industries).

References

Althoff, M., Frehse, G., & Girard, A. (2021). Set propagation techniques for reachability analysis. *Annual Review of Control, Robotics, and Autonomous Systems*, *4*(1).

Bakaráč, P., Holaza, J., Klaučo, M., Kalúz, M., Löfberg, J., & Kvasnica, M. (2018). Explicit MPC based on approximate dynamic programming. In *European control conference 2018*. Limassol, Cyprus: IEEE.

Bakaráč, P., Valiauga, P., & Kvasnica, M. (2018). Energy-efficient swing up and explicit MPC stabilization of an inverted pendulum. In *The 6th IFAC conference on nonlinear model predictive control*. Madison, Wisconsin, USA: IFAC.

Baotić, M., Borrelli, F., Bemporad, A., & Morari, M. (2008). Efficient on-line computation of constrained optimal control. *SIAM Journal on Control and Optimization*, *47*(5), 2470–2489.

Bayat, F., Johansen, T. A., & Jalali, A. A. (2011). Using hash tables to manage the time-storage complexity in a point location problem: Application to explicit model predictive control. *Automatica*, *47*(3), 571–577.

Bayat, F., Johansen, T. A., & Jalali, A. A. (2012). Flexible piecewise function evaluation methods based on truncated binary search trees and lattice representation in explicit MPC. *IEEE Transactions on Control Systems Technology*, *20*(3), 632–640.

Bemporad, A., Borrelli, F., & Morari, M. (2003). Min-max control of constrained uncertain discrete-time linear systems. *IEEE Transactions on Automatic Control*, *48*(9), 1600–1606.

Bemporad, A., Filippi, C., & Torrisi, F. D. (2004). Inner and outer approximation of polytopes using boxes. *Computational Geometry: Theory and Applications*, *27*(2), 151–178.

Bemporad, A., Morari, M., Dua, V., & Pistikopoulos, E. N. (2002). The explicit linear quadratic regulator for constrained systems. *Automatica*, *38*, 3–20.

Bemporad, A., Torrisi, F. D., & Morari, M. (2000). Optimization-based verification and stability characterization of piecewise affine and hybrid systems. In B. H. Krogh, & N. Lynch (Eds.), *Vol. 1790*, *International workshop on hybrid systems: computation and control* (pp. 45–58). Pittsburgh, USA: Springer-Verlag.

Bird, T. J., Jain, N., Pangborn, H. C., & Koeln, J. P. (2022). Set-based reachability and the explicit solution of linear MPC using hybrid zonotopes. In *2022 American control conference* (pp. 158–165).

Borrelli, F. (2017). *Constrained Optimal Control of Linear and Hybrid Systems*. Berlin, Heidelberg: Springer.

Borrelli, F., Baotić, M., Bemporad, A., & Morari, M. (2001). Efficient on-line computation of explicit model predictive control.

Borrelli, F., Baotić, M., Pekar, J., & Stewart, G. (2010). On the computation of linear model predictive control laws. *Automatica*, *46*(6), 1035–1041.

Borrelli, F., Bemporad, A., & Morari, M. (2017). *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press.

Changizi, N., Salahshoor, K., & Siahi, M. (2023). Design and implementation of a sub-optimal explicit mpc using a novel complexity reduction approach based on fuzzy reshaped active regions. *International Journal of Dynamics and Control*, *11*, 338–353.

Christophersen, F., Kvasnica, Jones, C., & Morari, M. (2007). Efficient evaluation of piecewise control laws defined over a large number of polyhedra. In *Proceedings of the European control conference*.

Fuchs, A., Jones, C., & Morari, M. (2010). Optimized decision trees for point location in polytopic data sets - application to explicit MPC. In *Proceedings of the 2010 American control conference* (pp. 5507–5512).

Geyer, T., Torrisi, F. D., & Morari, M. (2008). Optimal complexity reduction of polyhedral piecewise affine systems. *Automatica*, *44*(7), 1728–1740.

Gupta, A., Bhartiya, S., & Nataraj, P. S. V. (2011). A novel approach to multiparametric quadratic programming. *Automatica*, *47*(9), 2112–2117.

Herceg, M., Jones, C. N., Kvasnica, M., & Morari, M. (2015). Enumeration-based approach to solving parametric linear complementarity problems. *Automatica*, *62*, 243–248.

Herceg, M., Kvasnica, M., Jones, C., & Morari, M. (2013). Multi-parametric toolbox 3.0. In *2013 European control conference* (pp. 502–510).

Herceg, M., Mariéthoz, S., & Morari, M. (2013). Evaluation of piecewise affine control law via graph traversal. In *2013 European control conference* (pp. 3083–3088).

Holaza, J. (2012). *Complexity Reduction of Explicit Model Predictive Control*. Radlinského 9, 812 37 Bratislava: ÚIAM FCHPT STU v Bratislave.

Holaza, J., Oravec, J., Kvasnica, M., Dyrska, R., Mönnigmann, M., & Fikar, M. (2020). Accelerating explicit model predictive control by constraint sorting. In R. Findeisen, S. Hirche, K. Janschek, & M. Mönnigmann (Eds.), *The 21st IFAC world congress (virtual), Berlin, Germany* (pp. 11520–11525).

Holaza, J., Takács, B., Kvasnica, M., & Di Cairano, S. (2015a). Nearly optimal simple explicit MPC controllers with stability and feasibility guarantees. *Optimal Control Applications & Methods*, *35*(6).

Holaza, J., Takács, B., Kvasnica, M., & Di Cairano, S. (2015b). Safety verification of implicitly defined MPC feedback laws. In *European control conference 2015* (pp. 2552–2557). Linz, Austria.

Jafargholi, M., Peyrl, H., Zanarini, A., Herceg, M., & Mariethoz, S. (2014). Accelerating space traversal methods for explicit model predictive control via space partitioning trees. In *2014 European control conference* (pp. 103–108).

Johansen, T. A., & Grancharova, A. (2003). Approximate explicit constrained linear model predictive control via orthogonal search tree. *IEEE Transactions on Automatic Control*, *48*(5), 810–815.

Jones, C. N., & Morari, M. (2010). Polytopic approximation of explicit model predictive controllers. *IEEE Transactions on Automatic Control*, *55*.

Kvasnica, M., Bakaráč, P., & Klaučo, M. (2019). Complexity reduction in explicit MPC: A reachability approach. *Systems & Control Letters*, *124*, 19–26.

Kvasnica, M., & Fikar, M. (2010). Performance-lossless complexity reduction in Explicit MPC. In *49th IEEE conference on decision and control* (pp. 5270–5275).

Kvasnica, M., & Fikar, M. (2012). Clipping-based complexity reduction in explicit MPC. *IEEE Transactions on Automatic Control*, *57*(7), 1878–1883.

Kvasnica, M., Holaza, J., Takács, B., & Ingole, D. (2015). Design and verification of low-complexity explicit MPC controllers in MPT3. In *European control conference 2015* (pp. 2600–2605). Linz, Austria.

Kvasnica, M., Löfberg, J., & Fikar, M. (2011). Stabilizing polynomial approximation of explicit MPC. *Automatica*, *47*(10), 2292–2297.

Kvasnica, M., Takács, B., Holaza, J., & Ingole, D. (2015). Reachability analysis and control synthesis for uncertain linear systems in MPT. In M. Fikar (Ed.), *Vol. 8*, *Proceedings of the 8th IFAC symposium on robust control design* (pp. 302–307). Bratislava, Slovak Republic: Elsevier.

Mayne, D. Q. (2014). Model predictive control: Recent developments and future promise. *Automatica*, *50*, 2967–2986.

Mayne, D. Q., Seron, M. M., & Raković, S. V. (2005). Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, *41*(2), 219–224.

McInerney, I., Constantinides, G. A., & Kerrigan, E. C. (2018). A survey of the implementation of linear model predictive control on FPGAs. *IFAC-PapersOnLine*, *51*(20), 381–387, 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018.

Mitze, R., Kvasnica, M., & Mönnigmann, M. (2023). Exploiting symmetries in active set enumeration for constrained linear–quadratic optimal control. *Automatica*, *151*, Article 110900.

Morato, M. M., Normey-Rico, J. E., & Sename, O. (2020). Model predictive control design for linear parameter varying systems: A survey. *Annual Reviews in Control*, *49*, 64–80.

Nguyen, N. A. (2015). *Explicit robust constrained control for linear systems: analysis, implementation and design based on optimization* (Ph.D. thesis), CentraleSupélec, Université Paris Sud.

Oberdieck, R., Diangelakis, N. A., Nascu, I., Papathanasiou, M. M., Sun, M., Avraamidou, S., et al. (2016). On multi-parametric programming and its applications in process systems engineering. *Chemical Engineering Research and Design*, *116*, 61–82, Process Systems Engineering - A Celebration in Professor Roger Sargent's 90th Year.

Oberdieck, R., Diangelakis, N. A., & Pistikopoulos, E. N. (2017). Explicit model predictive control: A connected-graph approach. *Automatica*, *76*, 103–112.

Oravec, J., Jiang, Y., Houska, B., & Kvasnica, M. (2017). Parallel explicit MPC for hardware with limited memory. *Vol. 20*, In *The 20th IFAC world congress, Toulouse, France* (pp. 3356–3361).

Qin, S. J., & Badgwell, T. A. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*, *11*(7), 733–764.

Ramirez, D., & Camacho, E. (2006). Piecewise affinity of min-max MPC with bounded additive uncertainties and a quadratic criterion. *Automatica*, *42*(2), 295–302.

Spjøtvold, J., Raković, S. V., Tøndel, P., & Johansen, T. A. (2006). Utilizing reachability analysis in point location problems. In *Proceedings of the 45th IEEE conference on decision and control* (pp. 4568–4569).

Stursberg, O., & Krogh, B. H. (2003). Efficient representation and computation of reachable sets for hybrid systems. In O. Maler, & A. Pnueli (Eds.), *Hybrid systems: computation and control* (pp. 482–497). Berlin, Heidelberg: Springer.

Suardi, A., Longo, S., Kerrigan, E. C., & Constantinides, G. A. (2016). Explicit MPC: Hard constraint satisfaction under low precision arithmetic. *Control Engineering Practice*, *47*, 60–69.

Sui, D., Feng, L., & Hovd, M. (2008). Algorithms for online implementations of explicit MPC solutions. *IFAC Proceedings Volumes*, *41*(2), 3619–3622, 17th IFAC World Congress.

Tøndel, P., Johansen, T., & Bemporad, A. (2003). Evaluation of piecewise affine control via binary search tree. *Automatica*, *39*(5), 945–950.

Wang, Y., Jones, C., & Maciejowski, J. (2007). Efficient point location via subdivision walking with application to explicit MPC. In *2007 European control conference* (pp. 447–453).

Wen, C., Ma, X., & Ydstie, B. E. (2009). Analytical expression of explicit MPC solution via lattice piecewise-affine function. *Automatica*, *45*(4), 910–917.

Xu, J. (2021). Lattice piecewise affine approximation of explicit linear model predictive control. In *2021 60th IEEE conference on decision and control* (pp. 2545–2550).

Yan, D., Zhang, W., Chen, H., & Shi, J. (2023). Robust control strategy for multi-UAVs system using MPC combined with Kalman-consensus filter and disturbance observer. *ISA Transactions*, *135*, 35–51.

Zhang, J., & Xiu, X. (2018). K-d tree based approach for point location problem in explicit model predictive control. *Journal of the Franklin Institute*, *355*(13), 5431–5451.

Zhang, J., Xiu, X., Xie, Z., & Hu, B. (2016). Using a two-level structure to manage the point location problem in explicit model predictive control. *Asian Journal of Control*, *18*(3), 1075–1086.